ПРИЁМЫ ОБНАРУЖЕНИЯ ВИРТУАЛЬНЫХ МАШИН ПУТЁМ АНАЛИЗА ПАРАМЕТРОВ ОКРУЖЕНИЯ

Черемнов А.Г.
Национальный исследовательский Томский политехнический университет
8xandr@gmail.com

В настоящее время по всему миру постоянно увеличиваются объёмы создаваемого программного обеспечения. В след за этим растет количество нелегальных использований приложений и попыток его декомпиляции [1, 2]. Последнее неизбежно приводит к потере финансовой прибыли и раскрытию уникальных алгоритмов программного кода.

По этой причине разработка узкоспециализированных механизмов защиты программного кода с учётом аппаратных особенностей используемого оборудования является приоритетной задачей.

К основным инструментам хакеров, которые могут использоваться для облегчения процесса взлома приложений, относят [3, 4]:

- отладчики;
- дизассемблеры;
- редакторы ресурсов;
- распаковщики;
- редакторы PE-заголовка.

Не менее важным является и выбор среды исполнения для осуществления взлома. Огромное распространение здесь получили виртуальные машины, позволяющие достаточно быстро менять параметры среды окружения, операционной системы и оборудования в целом. Способность программы обнаружить своё исполнение внутри виртуальной машины и модифицировать свою работу может заметно усложнить процесс разбора программного кода.

Стоит отметить, что алгоритмы обнаружения виртуальных машин, используемые для защиты коммерческого приложения, являются коммерческой тайной компаний производителей программного обеспечения, в связи, с чем не описываются в литературе.

Как правило, существующие техники детектирования виртуальных машин основаны на анализе вспомогательных процессов, открытых окон приложений, объектов приложений, байт MAC-адреса, времени выполнения специальных инструкций, некорректном выполнении узкоспециализированных процессорных инструкций и атаках на конкретную виртуальную машину.

Анализ вспомогательных процессов коммерческим приложением обходится путём переименования исполняемых файлов или переименованием указанных процессов (таблица 1) в таблице процессов.

Таблица 1 – Анализ вспомогательных процессов

Виртуальная машина	Запущенные процессы
VirtualBox	VBoxTray.exe, VBoxService.exe
Parallels Workstation	prl_cc.exe, prl_tools.exe, SharedIntApp.exe
Virtual PC	vmusrvc.exe, vmsrvc.exe
VMware Workstation	vmtoolsd.exe

Помимо вспомогательных процессов можно анализировать и объекты, порождаемые этими процессами в ОЗУ компьютера (таблица 2).

Таблица 2 – Анализ объектов вспом. приложений

Виртуальная машина	Название объектов
VirtualBox	DeviceVBoxMiniRdrDN, DeviceVBoxGuest
Parallels Workstation	Deviceprl_pv, Deviceprl_tg, Deviceprl_time, DevicePrlMemDev
Virtual PC	DeviceVirtualMachineServices DeviceVirtualMachineServicesPCI DeviceVirtualMachineServicesUSB
VMware Workstation	DevicePrlMemDevPci, DevicePrlMemDev

Идентификация виртуальной машины в этом методе основана на попытке создания объекта с именем, приведенным в таблице 2. Данная операция осуществляется при помощи функции WinAPI CreateFile. Ошибка создания объекта является признаком существования объекта. Обойти подобную проверку возможно перехватом вызова функции CreateFile или непосредственной модификацией системной динамической библиотеки Microsoft Windows - kernel32.dll.

Обход анализа открытых окон приложений (таблица 3) осуществляется при помощи перехвата WinAPI функций или переименованием окон, используя методы WinAPI до запуска исследуемого приложения.

Таблица 4 – Открытые окна

Виртуальная машина	Название объектов
VirtualBox	VBoxTrayToolWndClass
Parallels Workstation	CPInterceptor, DesktopUtilites
Virtual PC	{0843FD01-1D28-44a3-B11D-E3A93A85EA96}
VMware Workstation	VMSwitchUserControlClass

Анализ первых трёх байт MAC-адреса виртуального сетевого адаптера (таблица 4), обеспечивающего соединение гостевой виртуальной машиной с хостовой системой, также не обеспечивает необходимой защиты - параметры сетевого интерфейса легко изменить.

Таблица 4 – Статичная часть MAC адреса

Виртуальная машина	Статичная часть физического адреса
VirtualBox	080020h, 080026h, 080024h, 08002Dh, 08002Eh, 080027h
Parallels Workstation	001C42h
Virtual PC	0003FFh, 000D3Ah, 0050F2h, 7C1E52h, 00125Ah, 00155Dh, 0017FAh, 281878h, 7CED8Dh, 001DD8h, 6045BDh, DCB4C4h
VMware Workstation	001C42h

Анализ времени выполнения инструкций является более надёжным подходом, хотя он уже и не работает на VMware Workstation 12-ой версии. Пример исходного кода на языке Delphi:

```
function IsVm(): dword; register;
var
k: DWORD;
asm
  push ebx
  push edi
  @@r1:
  db $0f, $31
  mov edi, edx
  mov ebx, eax
  db $0f, $31
  cmp edi, edx
  jnz @@r1
  sub eax, ebx
  mov k, eax
  mov ecx, $0a
  @@cycle:
  db $0f, $31
  mov edi, edx
  mov ebx, eax
  db $0f, $31
  cmp edi, edx
  jnz @@cycle
  sub eax, ebx
  cmp eax, k
  jg @@ext1
  mov k, eax
@@ext1:
  dec ecx
  jnz @@cycle
  mov eax, k
  pop edi
  pop ebx
end;
function IsVmTest01: boolean;
begin
  Result := FALSE;
  if (IsVm > 200) then
  Result := TRUE;
end;
```

Однако, используя Windows XP, VMware Workstation можно идентифицировать по следующему алгоритму [5]:

1. Записать в регистр EAX -> 564d5868h;
2. Записать в регистр ECX -> 0Ah;
3. Прочитать данные из порта 5658h;
4. Выполнение вернёт текущую версию виртуальной машины, в данном случае 12.

Последовательный вызов опкодов 0Fh, 3Fh, 07h и 0Bh на реальном процессоре приводит к возникновению ошибки времени выполнения, но на эмулированном процессоре VirtualPC выполняется без ошибок, что позволяет явно идентифицировать эту виртуальную машину.

Пример исходного кода атаки RedPill:

```
function SwallowRedpill: Boolean;
var
  RedPill: array[0..7] of byte;
  m: array[0..5] of byte;
  p: procedure; stdcall;
begin
  RedPill[0]:= $0f; RedPill[1]:=
$01; RedPill[2]:= $0d; RedPill[3]:=
$00;
  RedPill[4]:= $00; RedPill[5]:=
$00; RedPill[6]:= $00; RedPill[7]:=
$c3;
  // Установка указателя
  PPointer(@RedPill[3])^:= @m;
  // Возврат адреса
  p:= @RedPill;
  // Вызов p
  p();
  // Детектирование ВМ
  Result:= m[5] > $d0; end;
```

В результате работы реализован модуль детектирования виртуальных машин, который в настоящий момент используется для защиты программного обеспечения для восстановления информации с жестких дисков.

Литература

1. Даррелл Пейнетьер. Живучесть пиратства и его последствия для творчества, культуры и устойчивого развития. – Париж: UNESCO, 2005 – 21 с.
2. Sixth Annual BSA and IDC Global Software Piracy Study. URL: http://global.bsa.org/globalpiracy2008/studies/globalpiracy2008.pdf (Дата обращения: 20.10.2015)
3. Грег Хогланд, Гари Мак-Гроу. Взлом программного обеспечения: анализ и использолвание программного кода. – М.: Вильямс, 2005 г. – 384 с.
4. Крис Касперский. Фундаментальные основы хакерства. Искусство дизассемблирования. М: СОЛОН-Р, 2002 г. – 448 с.